CLAIMS

We claim:

1. A method of compressing a computer program, comprising the steps of:

scanning an initial computer program to identify a first plurality of uncompressed instructions therein having a high frequency of use;

populating a first storage mechanism with the identified first plurality of uncompressed instructions; and

generating a first compressed computer program by replacing each of a plurality of the identified first plurality of uncompressed instructions in the initial computer program with a respective first type of compressed instruction that identifies a location of the corresponding uncompressed instruction in the first storage mechanism.

2. A method as recited in Claim 1, further comprising the steps of:

scanning the first compressed computer program to identify a second plurality of uncompressed instructions that have a high frequency of use when at least a portion of their respective instruction operand is ignored;

populating a second storage mechanism with the identified second plurality of uncompressed instructions; and

generating a second compressed computer program by replacing each of a plurality of the identified second plurality of uncompressed instructions in the first compressed computer program with a respective second type of compressed instruction that identifies a location of the corresponding uncompressed instruction in the second storage mechanism.

3. A method as recited in Claim 1, further comprising the step of:

identifying addresses referenced in the initial computer program that are used by instructions that transfer control before the step of generating the first compressed computer program.

4. A method as recited in Claim 3, further comprising the steps of:

calculating new addresses for the first compressed computer program instructions;

determining if the identified addresses that are used by instructions that transfer control have changed in response to the step of calculating new addresses for the first compressed computer program instructions; and

updating each identified address that is referenced in the first compressed computer program and has changed with the calculated new address that corresponds thereto.

5. A method as recited in Claim 4, wherein the step of updating each identified address that is referenced in the first compressed computer program and has changed with the calculated new address that corresponds thereto, comprises the steps of:

storing the identified addresses that are used by instructions that transfer control in a second storage mechanism;

associating each identified address that has changed with the calculated new address that corresponds thereto in the second storage mechanism; and

updating each identified address that has changed with the calculated new address that is associated therewith in the second storage mechanism.

6. A method as recited in Claim 1, wherein the computer program instructions follow a format in which at least two bits are used to define the instruction type.

7. A method as recited in Claim 1, wherein the first storage mechanism and the second storage mechanism comprise a single data structure.

8. A method as recited in Claim 1, wherein the first storage mechanism and the second storage mechanism comprise separate data structures.

9. A method as recited in Claim 1, wherein the step of scanning an initial computer program to identify a first plurality of uncompressed instructions therein having a high frequency of use comprises the step of:

scanning the initial computer program to identify the first plurality of uncompressed instructions therein having a high frequency of use based on a non-operand portion thereof; and wherein the method further comprises the steps of:

scanning the initial computer program to identify a first plurality of operands therein having a high frequency of use; and

populating a second storage mechanism with the identified first plurality of operands.

10. A method of compressing a computer program, comprising the steps of:

scanning an initial computer program to identify a first plurality of uncompressed instructions therein based on a first compression criterion;

populating a first storage mechanism with the identified first plurality of uncompressed instructions; and

generating a first compressed computer program by replacing each of a plurality of the identified first plurality of uncompressed instructions in the initial computer program with a respective first compressed instruction that identifies a location of the corresponding uncompressed instruction in the first storage mechanism.

11. A method as recited in Claim 10, further comprising the steps of:

scanning the first compressed computer program to identify a second plurality of uncompressed instructions based on a second compression criterion;

populating a second storage mechanism with the identified second plurality of uncompressed instructions; and

generating a second compressed computer program by replacing each of a plurality of the identified second plurality of uncompressed instructions in the first compressed computer program with a respective second compressed instruction that identifies a location of the corresponding uncompressed instruction in the second storage mechanism.

12. A method as recited in Claim 11, wherein the first compression criterion is instruction frequency of use and wherein the second compression criterion

is instruction frequency of use when at least a portion of their respective instruction operand is ignored.

13. A method as recited in Claim 10, wherein the first compression criterion is instruction execution speed.

14. A method of compressing a computer program, comprising the steps of:

respectively scanning each of a plurality of routines in an initial computer program to identify a first plurality of uncompressed instructions in each of the plurality of routines that have a high frequency of use;

respectively populating first storage mechanisms with the identified first plurality of uncompressed instructions from each of the plurality of routines; and

generating a first compressed computer program by respectively replacing each of a plurality of the identified first plurality of uncompressed instructions in each of the plurality of routines with a respective first compressed instruction that identifies a location of the corresponding uncompressed instruction in a respective one of the first storage mechanisms.

15. A method as recited in Claim 14, further comprising the steps of:

respectively scanning each of the plurality of routines in the first compressed computer program to identify a second plurality of uncompressed instructions in each of the plurality of routines that have a high frequency of use;

respectively populating second storage mechanisms with the identified second plurality of uncompressed instructions from each of the plurality of routines; and

generating a second compressed computer program by respectively replacing each of a plurality of the identified second plurality of uncompressed instructions in each of the plurality of routines with a respective second compressed instruction that identifies a location of the corresponding uncompressed instruction in a respective one of the second storage mechanisms.

16. A method of executing a computer program, comprising the steps of:

fetching an instruction from a memory;

decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction, a first type of compressed instruction, a second type of compressed instruction, or a third type of compressed instruction;

decoding the fetched instruction to identify a location in a first logical data structure, if the fetched instruction is a compressed instruction of the first type;

providing a first uncompressed instruction, which is located at the location in the first logical data structure, to a processor for execution if the fetched instruction is a compressed instruction of the first type;

decoding the fetched instruction to identify a location in a second logical data structure, if the fetched instruction is a compressed instruction of the second type;

combining portions of the fetched instruction with portions of an at least partially uncompressed instruction, which is located at the location in the second logical data structure, to generate a second uncompressed instruction if the fetched instruction is a compressed instruction of the second type;

providing the second uncompressed instruction to the processor for execution if the fetched instruction is a compressed instruction of the second type;

decoding the fetched instruction to identify a location in a third logical data structure, if the fetched instruction is a compressed instruction of the third type;

decoding the fetched instruction to identify a location in an operand data structure, if the fetched instruction is a compressed instruction of the third type;

combining a non-operand portion of an uncompressed instruction, which is located at the location in the third logical data structure, with an operand portion of the uncompressed instruction, which is located at the location in the operand data structure, to generate a third uncompressed instruction if the fetched instruction is a compressed instruction of the third type; and

providing the third uncompressed instruction to the processor for execution, if the fetched instruction is a compressed instruction of the third type.

17. A method as recited in Claim 16, further comprising the steps of:

downloading the first logical data structure from the memory to a first decompression sub-engine before the step of decoding the fetched instruction to identify a location in the first logical data structure;

downloading the second logical data structure from the memory to a second decompression sub-engine before the step of decoding the fetched instruction to identify a location in the second logical data structure;

downloading the third logical data structure and the operand data structure from the memory to a third decompression sub-engine before the steps of decoding the fetched instruction to identify a location in the third logical data structure and decoding the fetched instruction to identify a location in the operand data structure;

providing the fetched instruction to the first decompression sub-engine if the fetched instruction is a compressed instruction of the first type before the step of decoding the fetched instruction to identify a location in the first logical data structure;

providing the fetched instruction to the second decompression sub-engine if the fetched instruction is a compressed instruction of the second type before the step of decoding the fetched instruction to identify a location in the second logical data structure; and

providing the fetched instruction to the third decompression sub-engine if the fetched instruction is a compressed instruction of the third type before the steps of decoding the fetched instruction to identify a location in the third logical data structure and decoding the fetched instruction to identify a location in the operand data structure.

18. A method as recited in Claim 16, wherein the first logical data structure and the second logical data structure comprise a single data structure.

19. A method as recited in Claim 16, wherein the first logical data structure and the second logical data structure comprise separate data structures.

20. A method of executing a computer program, comprising the steps of:

fetching an instruction associated with one of a plurality of routines from a memory;

decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction or a first type of compressed instruction;

decoding the fetched instruction to identify a location in a first logical data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the first type; and

providing a first uncompressed instruction, which is located at the location in the first logical data structure, to a processor for execution if the fetched instruction is a compressed instruction of the first type.

21. A method as recited in Claim 20, wherein the step of decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction or a first type of compressed instruction comprises the step of:

decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction, a first type of compressed instruction, a second type of compressed instruction, or a third type of compressed instruction.

22. A method as recited in Claim 21, further comprising the steps of:

decoding the fetched instruction to identify a location in a second logical data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the second type;

combining portions of the fetched instruction with portions of an at least partially uncompressed instruction, which is located at the location in the second logical data structure, to generate a second uncompressed instruction if the fetched instruction is a compressed instruction of the second type;

providing the second uncompressed instruction to the processor for execution if the fetched instruction is a compressed instruction of the second type;

decoding the fetched instruction to identify a location in a third logical data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the third type;

decoding the fetched instruction to identify a location in an operand data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the third type;

combining a non-operand portion of an uncompressed instruction, which is located at the location in the third logical data structure, with an operand portion of the uncompressed instruction, which is located at the location in the operand data

structure, to generate a third uncompressed instruction if the fetched instruction is a compressed instruction of the third type; and

providing the third uncompressed instruction to the processor for execution, if the fetched instruction is a compressed instruction of the third type.

23. A method as recited in Claim 22, wherein the first logical data structure and the second logical data structure comprise a single data structure.

24. A method as recited in Claim 22, wherein the first logical data structure and the second logical data structure comprise separate data structures.

25. A method of operating a decompression unit for compressed computer program instructions, comprising the steps of:

loading into a buffer an instruction from a memory;

receiving an instruction address from a processor;

determining if the instruction address corresponds to a sequential instruction;

decoding the instruction in the buffer to determine whether the decoded instruction is a compressed instruction or an uncompressed instruction if the instruction address corresponds to a sequential instruction; and

removing the decoded instruction from the buffer.

26. A method as recited in Claim 25, further comprising the steps of:

decompressing the decoded instruction if the decoded instruction is a compressed instruction; and

providing the decompressed decoded instruction at an output of the decompression unit.

27. A method as recited in Claim 25, wherein the step of determining if the instruction address corresponds to a sequential instruction comprises the step of:

comparing the instruction address from the processor with a previous instruction address.

28. A method as recited in Claim 25, wherein the buffer is selected from the group consisting of a latch and a register.

29. A method as recited in Claim 28, wherein buffer has a length of at least two uncompressed instructions, and wherein the step of removing the decoded instruction from the buffer comprises the step of:

shifting the buffer an amount corresponding to a length of the decoded instruction.

30. A method as recited in Claim 25, further comprising the steps of:

clearing the buffer of its contents if the instruction address does not correspond to a sequential instruction;

loading into the buffer an instruction that is located in the memory at the received instruction address; and

decoding the instruction that is located in the memory at the received instruction address and has been loaded into the buffer to determine whether the decoded instruction is a compressed instruction or an uncompressed instruction.

31. A data processing system for decompressing compressed computer program instructions, comprising:

an instruction type decoding unit having a data input that receives an instruction and determines whether the received instruction is an uncompressed instruction, a first type of compressed instruction, a second type of compressed instruction, or a third type of compressed instruction;

a first decompression sub-engine for the first type of compressed instruction having a data input coupled to a first data output of the instruction type decoding unit; and

a second decompression sub-engine for the second type of compressed instruction having a data input coupled to a second data output of the instruction type decoding unit.

32. A data processing system as recited in Claim 31, further comprising:

a third decompression sub-engine for the third type of compressed instruction having a data input coupled to a third data output of the instruction type decoding unit.

33. A data processing system as recited in Claim 32, wherein the first decompression sub-engine comprises a first memory that is configured with a first data structure in which compressed instructions of the first type are respectively associated with first uncompressed instructions.

34. A data processing system as recited in Claim 33, wherein the second decompression sub-engine comprises a second memory that is configured with a second data structure in which compressed instructions of the second type are respectively associated with second at least partially uncompressed instructions.

35. A data processing system as recited in Claim 34, wherein the third decompression sub-engine comprises a third memory that is configured with a third data structure in which compressed instructions of the third type are respectively associated with third at least partially uncompressed instructions.

36. A data processing system as recited in Claim 32, wherein the first, second, and third decompression sub-engines are communicatively coupled to a main memory, and wherein the instruction type decoding unit further comprises:

a data structure load unit that is configured to detect a data structure load instruction and to facilitate downloading the first, second, and third data structures to the first, second, and third decompression sub-engines, respectively.

37. A data processing system as recited in Claim 32, further comprising:

a multiplexer having a first data input coupled to a third data output of the instruction type decoding unit, a second data input coupled to a data output of the first decompression sub-engine, a third data input coupled to a data output of the second decompression sub-engine, a fourth data input coupled to a data output of the third decompression sub-engine, and a select input that receives a select signal generated by the instruction type decoding unit.

38. A data processing system as recited in Claim 31, further comprising:

an address translation unit having a data input that receives an instruction address from a processor and that generates in response thereto a jump signal if the instruction address is indicative of a transfer of control, that generates a sequential signal if the instruction address is indicative of sequential instruction execution;

a memory fetch unit having first, second, and third data inputs for receiving the jump signal, the sequential signal, and the instruction address from the address translation unit and a data output that is communicatively coupled to a main memory; and

a current address register that is communicatively coupled to the memory fetch unit and the instruction type decoding unit and that contains an address of the received instruction.

39. A data processing system as recited in Claim 31, further comprising:

a buffer having a data input that is communicatively coupled to the main memory for receiving the received instruction from the main memory, a data output that is coupled to the data input of the instruction type decoding unit, and a reset input that is coupled to the jump signal.

40. A system for compressing a computer program, comprising:

means for scanning an initial computer program to identify a first plurality of uncompressed instructions therein having a high frequency of use;

means for populating a first storage mechanism with the identified first plurality of uncompressed instructions; and

means for generating a first compressed computer program by replacing each of a plurality of the identified first plurality of uncompressed instructions in the initial computer program with a respective first type of compressed instruction that identifies a location of the corresponding uncompressed instruction in the first storage mechanism.

41. A system as recited in Claim 40, further comprising:

means for scanning the first compressed computer program to identify a second plurality of uncompressed instructions that have a high frequency of use when at least a portion of their respective instruction operand is ignored;

means for populating a second storage mechanism with the identified second plurality of uncompressed instructions; and

means for generating a second compressed computer program by replacing each of a plurality of the identified second plurality of uncompressed instructions in the first compressed computer program with a respective second type of compressed instruction that identifies a location of the corresponding uncompressed instruction in the second storage mechanism.

42. A system as recited in Claim 40, further comprising:

means for identifying addresses referenced in the initial computer program that are used by instructions that transfer control, the means for generating the first compressed computer program being responsive to the means for identifying addresses.

43. A system as recited in Claim 42, further comprising:

means for calculating new addresses for the first compressed computer program instructions;

means for determining if the identified addresses that are used by instructions that transfer control have changed, the means for determining being responsive to the means for calculating new addresses for the first compressed computer program instructions; and

means for updating each identified address that is referenced in the first compressed computer program and has changed with the calculated new address that corresponds thereto.

44. A system as recited in Claim 43, wherein the means for updating each identified address that is referenced in the first compressed computer program and has changed with the calculated new address that corresponds thereto, comprises:

means for storing the identified addresses that are used by instructions that transfer control in a second storage mechanism;

means for associating each identified address that has changed with the calculated new address that corresponds thereto in the second storage mechanism; and

means for updating each identified address that has changed with the calculated new address that is associated therewith in the second storage mechanism.

45. A system as recited in Claim 40, wherein the computer program instructions follow a format in which at least two bits are used to define the instruction type.

46. A system as recited in Claim 40, wherein the first storage mechanism and the second storage mechanism comprise a single data structure.

47. A system as recited in Claim 40, wherein the first storage mechanism and the second storage mechanism comprise separate data structures.

48. A system for compressing a computer program, comprising:

means for scanning an initial computer program to identify a first plurality of uncompressed instructions therein based on a first compression criterion;

means for populating a first storage mechanism with the identified first plurality of uncompressed instructions; and

means for generating a first compressed computer program by replacing each of a plurality of the identified first plurality of uncompressed instructions in the initial computer program with a respective first compressed instruction that identifies a location of the corresponding uncompressed instruction in the first storage mechanism.

49. A system as recited in Claim 48, further comprising:

means for scanning the first compressed computer program to identify a second plurality of uncompressed instructions based on a second compression criterion;

means for populating a second storage mechanism with the identified second plurality of uncompressed instructions; and

means for generating a second compressed computer program by replacing each of a plurality of the identified second plurality of uncompressed instructions in the first compressed computer program with a respective second compressed instruction that identifies a location of the corresponding uncompressed instruction in the second storage mechanism.

50. A system as recited in Claim 49, wherein the first compression criterion is instruction frequency of use and wherein the second compression criterion is instruction frequency of use when at least a portion of their respective instruction operand is ignored.

51. A system as recited in Claim 48, wherein the first compression criterion is instruction execution speed.

52. A system for executing a computer program, comprising:

means for fetching an instruction from a memory;

means for decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction, a first type of compressed instruction, or a second type of compressed instruction;

means for decoding the fetched instruction to identify a location in a first logical data structure, if the fetched instruction is a compressed instruction of the first type;

means for providing a first uncompressed instruction, which is located at the location in the first logical data structure, to a processor for execution if the fetched instruction is a compressed instruction of the first type;

means for decoding the fetched instruction to identify a location in a second logical data structure, if the fetched instruction is a compressed instruction of the second type;

means for combining portions of the fetched instruction with portions of an at least partially uncompressed instruction, which is located at the location in the second logical data structure, to generate a second uncompressed instruction if the fetched instruction is a compressed instruction of the second type; and

means for providing the second uncompressed instruction to the processor for execution if the fetched instruction is a compressed instruction of the second type;

means for decoding the fetched instruction to identify a location in a third logical data structure, if the fetched instruction is a compressed instruction of the third type;

means for decoding the fetched instruction to identify a location in an operand data structure, if the fetched instruction is a compressed instruction of the third type;

means for combining a non-operand portion of an uncompressed instruction, which is located at the location in the third logical data structure, with an operand portion of the uncompressed instruction, which is located at the location in the operand data structure, to generate a third uncompressed instruction if the fetched instruction is a compressed instruction of the third type; and

means for providing the third uncompressed instruction to the processor for execution, if the fetched instruction is a compressed instruction of the third type.

53. A system as recited in Claim 52, further comprising:

means for downloading the first logical data structure from the memory to a first decompression sub-engine, the means for decoding the fetched instruction to identify a location in the first logical data structure being responsive to the means for downloading the first logical data structure;

means for downloading the second logical data structure from the memory to a second decompression sub-engine, the means for decoding the fetched instruction to identify a location in the second logical data structure being responsive to the means for downloading the second logical data structure;

means for downloading the third logical data structure and the operand data structure from the memory to a third decompression sub-engine, the means for decoding the fetched instruction to identify a location in the third logical data structure and the means for decoding the fetched instruction to identify a location in the operand data structure being responsive to the means for downloading the third logical data structure;

means for providing the fetched instruction to the first decompression sub-engine if the fetched instruction is a compressed instruction of the first type that is

responsive to the means for decoding the fetched instruction to identify a location in the first logical data structure;

means for providing the fetched instruction to the second decompression sub-engine if the fetched instruction is a compressed instruction of the second type that is responsive to the means for decoding the fetched instruction to identify a location in the second logical data structure; and

means for providing the fetched instruction to the third decompression sub-engine if the fetched instruction is a compressed instruction of the third type before the steps of decoding the fetched instruction to identify a location in the third logical data structure and decoding the fetched instruction to identify a location in the operand data structure.

54. A system as recited in Claim 52, wherein the first logical data structure and the second logical data structure comprise a single data structure.

55. A system as recited in Claim 52, wherein the first logical data structure and the second logical data structure comprise separate data structures.

56. A system for executing a computer program, comprising:

means for fetching an instruction associated with one of a plurality of routines from a memory;

means for decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction or a first type of compressed instruction;

means for decoding the fetched instruction to identify a location in a first logical data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the first type; and

means for providing a first uncompressed instruction, which is located at the location in the first logical data structure, to a processor for execution if the fetched instruction is a compressed instruction of the first type.

57. A system as recited in Claim 56, wherein the means for decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction or a first type of compressed instruction comprises:

means for decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction, a first type of compressed instruction, a second type of compressed instruction, or a third type of compressed instruction.

58. A system as recited in Claim 57, further comprising:

means for decoding the fetched instruction to identify a location in a second logical data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the second type;

means for combining portions of the fetched instruction with portions of an at least partially uncompressed instruction, which is located at the location in the second logical data structure, to generate a second uncompressed instruction if the fetched instruction is a compressed instruction of the second type;

means for providing the second uncompressed instruction to the processor for execution if the fetched instruction is a compressed instruction of the second type;

means for decoding the fetched instruction to identify a location in a third logical data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the third type;

means for decoding the fetched instruction to identify a location in an operand data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the third type;

means for combining a non-operand portion of an uncompressed instruction, which is located at the location in the third logical data structure, with an operand portion of the uncompressed instruction, which is located at the location in the operand data structure, to generate a third uncompressed instruction if the fetched instruction is a compressed instruction of the third type; and

means for providing the third uncompressed instruction to the processor for execution, if the fetched instruction is a compressed instruction of the third type.

59. A system as recited in Claim 58, wherein the first logical data structure and the second logical data structure comprise a single data structure.

60. A system as recited in Claim 58, wherein the first logical data structure and the second logical data structure comprise separate data structures.

61. A computer program product for compressing a computer program, comprising:

a computer readable storage medium having computer readable program code embodied therein, the computer readable program code comprising:

computer readable program code for scanning an initial computer program to identify a first plurality of uncompressed instructions therein having a high frequency of use;

computer readable program code for populating a first storage mechanism with the identified first plurality of uncompressed instructions; and

computer readable program code for generating a first compressed computer program by replacing each of a plurality of the identified first plurality of uncompressed instructions in the initial computer program with a respective first type of compressed instruction that identifies a location of the corresponding uncompressed instruction in the first storage mechanism.

62. A computer program product as recited in Claim 61, further comprising:

computer readable program code for scanning the first compressed computer program to identify a second plurality of uncompressed instructions that have a high frequency of use when at least a portion of their respective instruction operand is ignored;

computer readable program code for populating a second storage mechanism with the identified second plurality of uncompressed instructions; and

computer readable program code for generating a second compressed computer program by replacing each of a plurality of the identified second plurality of uncompressed instructions in the first compressed computer program with a respective second type of compressed instruction that identifies a location of the corresponding uncompressed instruction in the second storage mechanism.

63. A computer program product as recited in Claim 61, further comprising:

computer readable program code for identifying addresses referenced in the initial computer program that are used by instructions that transfer control, the computer readable program code for generating the first compressed computer program being responsive to the computer readable program code for identifying addresses.

64. A computer program product as recited in Claim 63, further comprising:

computer readable program code for calculating new addresses for the first compressed computer program instructions;

computer readable program code for determining if the identified addresses that are used by instructions that transfer control have changed, the computer readable program code for determining being responsive to the computer readable program code for calculating new addresses for the first compressed computer program instructions; and

computer readable program code for updating each identified address that is referenced in the first compressed computer program and has changed with the calculated new address that corresponds thereto.

65. A computer program product as recited in Claim 64, wherein the computer readable program code for updating each identified address that is referenced in the first compressed computer program and has changed with the calculated new address that corresponds thereto, comprises:

computer readable program code for storing the identified addresses that are used by instructions that transfer control in a second storage mechanism;

computer readable program code for associating each identified address that has changed with the calculated new address that corresponds thereto in the second storage mechanism; and

computer readable program code for updating each identified address that has changed with the calculated new address that is associated therewith in the second storage mechanism.

66. A computer program product as recited in Claim 61, wherein the computer program instructions follow a format in which at least two bits are used to define the instruction type.

67. A computer program product as recited in Claim 61, wherein the first storage mechanism and the second storage mechanism comprise a single data structure.

68. A computer program product as recited in Claim 61, wherein the first storage mechanism and the second storage mechanism comprise separate data structures.

69. A computer program product for compressing a computer program, comprising:

a computer readable storage medium having computer readable program code embodied therein, the computer readable program code comprising:

computer readable program code for scanning an initial computer program to identify a first plurality of uncompressed instructions therein based on a first compression criterion;

computer readable program code for populating a first storage mechanism with the identified first plurality of uncompressed instructions; and

computer readable program code for generating a first compressed computer program by replacing each of a plurality of the identified first plurality of uncompressed instructions in the initial computer program with a respective first compressed instruction that identifies a location of the corresponding uncompressed instruction in the first storage mechanism.

70. A computer program product as recited in Claim 69, further comprising:

computer readable program code for scanning the first compressed computer program to identify a second plurality of uncompressed instructions based on a second compression criterion;

computer readable program code for populating a second storage mechanism with the identified second plurality of uncompressed instructions; and

computer readable program code for generating a second compressed computer program by replacing each of a plurality of the identified second plurality of uncompressed instructions in the first compressed computer program with a respective second compressed instruction that identifies a location of the corresponding uncompressed instruction in the second storage mechanism.

71. A computer program product as recited in Claim 70, wherein the first compression criterion is instruction frequency of use and wherein the second compression criterion is instruction frequency of use when at least a portion of their respective instruction operand is ignored.

72. A computer program product as recited in Claim 69, wherein the first compression criterion is instruction execution speed.

73. A computer program product for executing a computer program, comprising:

a computer readable storage medium having computer readable program code embodied therein, the computer readable program code comprising:

computer readable program code for fetching an instruction from a memory;

computer readable program code for decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction, a first type of compressed instruction, or a second type of compressed instruction;

computer readable program code for decoding the fetched instruction to identify a location in a first logical data structure, if the fetched instruction is a compressed instruction of the first type;

computer readable program code for providing a first uncompressed instruction, which is located at the location in the first logical data structure, to a processor for execution if the fetched instruction is a compressed instruction of the first type;

computer readable program code for decoding the fetched instruction to identify a location in a second logical data structure, if the fetched instruction is a compressed instruction of the second type;

computer readable program code for combining portions of the fetched instruction with portions of an at least partially uncompressed instruction, which is located at the location in the second logical data structure, to generate a second uncompressed instruction if the fetched instruction is a compressed instruction of the second type; and

computer readable program code for providing the second uncompressed instruction to the processor for execution if the fetched instruction is a compressed instruction of the second type;

computer readable program code for decoding the fetched instruction to identify a location in a third logical data structure, if the fetched instruction is a compressed instruction of the third type;

computer readable program code for decoding the fetched instruction to identify a location in an operand data structure, if the fetched instruction is a compressed instruction of the third type;

computer readable program code for combining a non-operand portion of an uncompressed instruction, which is located at the location in the third logical data structure, with an operand portion of the uncompressed instruction, which is located at the location in the operand data structure, to generate a third uncompressed instruction if the fetched instruction is a compressed instruction of the third type; and

computer readable program code for providing the third uncompressed instruction to the processor for execution, if the fetched instruction is a compressed instruction of the third type.

74. A computer program product as recited in Claim 73, further comprising:

computer readable program code for downloading the first logical data structure from the memory to a first decompression sub-engine, the computer readable program code for decoding the fetched instruction to identify a location in the first logical data structure being responsive to the computer readable program code for downloading the first logical data structure;

computer readable program code for downloading the second logical data structure from the memory to a second decompression sub-engine, the computer readable program code for decoding the fetched instruction to identify a location in the second logical data structure being responsive to the computer readable program code for downloading the second logical data structure;

computer readable program code for downloading the third logical data structure and the operand data structure from the memory to a third decompression sub-engine, the computer readable program code for decoding the fetched instruction to identify a location in the third logical data structure and the computer readable program code for decoding the fetched instruction to identify a location in the operand data structure being responsive to the computer readable program code for downloading the third logical data structure;

computer readable program code for providing the fetched instruction to the first decompression sub-engine if the fetched instruction is a compressed instruction of the first type that is responsive to the computer readable program code for decoding the fetched instruction to identify a location in the first logical data structure;

computer readable program code for providing the fetched instruction to the second decompression sub-engine if the fetched instruction is a compressed instruction of the second type that is responsive to the computer readable program code for decoding the fetched instruction to identify a location in the second logical data structure; and

computer readable program code for providing the fetched instruction to the third decompression sub-engine if the fetched instruction is a compressed instruction of the third type before the steps of decoding the fetched instruction to identify a location in the third logical data structure and decoding the fetched instruction to identify a location in the operand data structure.

75. A computer program product as recited in Claim 73, wherein the first logical data structure and the second logical data structure comprise a single data structure.

76. A computer program product as recited in Claim 73, wherein the first logical data structure and the second logical data structure comprise separate data structures.

77. A computer program product for executing a computer program, comprising:

a computer readable storage medium having computer readable program code embodied therein, the computer readable program code comprising:

computer readable program code for fetching an instruction associated with one of a plurality of routines from a memory;

computer readable program code for decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction or a first type of compressed instruction;

computer readable program code for decoding the fetched instruction to identify a location in a first logical data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the first type; and

computer readable program code for providing a first uncompressed instruction, which is located at the location in the first logical data structure, to a processor for execution if the fetched instruction is a compressed instruction of the first type.

78. A computer program product as recited in Claim 77, wherein the computer readable program code for decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction or a first type of compressed instruction comprises:

computer readable program code for decoding the fetched instruction to determine whether the fetched instruction is an uncompressed instruction, a first type of compressed instruction, a second type of compressed instruction, or a third type of compressed instruction.

79. A computer program product as recited in Claim 78, further comprising:

computer readable program code for decoding the fetched instruction to identify a location in a second logical data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the second type;

computer readable program code for combining portions of the fetched instruction with portions of an at least partially uncompressed instruction, which is located at the location in the second logical data structure, to generate a second uncompressed instruction if the fetched instruction is a compressed instruction of the second type;

computer readable program code for providing the second uncompressed instruction to the processor for execution if the fetched instruction is a compressed instruction of the second type;

computer readable program code for decoding the fetched instruction to identify a location in a third logical data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the third type;

computer readable program code for decoding the fetched instruction to identify a location in an operand data structure that is exclusively associated with the one of the plurality of routines, if the fetched instruction is a compressed instruction of the third type;

computer readable program code for combining a non-operand portion of an uncompressed instruction, which is located at the location in the third logical data structure, with an operand portion of the uncompressed instruction, which is located at the location in the operand data structure, to generate a third uncompressed instruction if the fetched instruction is a compressed instruction of the third type; and

computer readable program code for providing the third uncompressed instruction to the processor for execution, if the fetched instruction is a compressed instruction of the third type.

80. A computer program product as recited in Claim 79, wherein the first logical data structure and the second logical data structure comprise a single data structure.

81. A computer program product as recited in Claim 79, wherein the first logical data structure and the second logical data structure comprise separate data structures.